Ruby trunk - Feature #13568

File#path for O_TMPFILE fds has no meaning

05/15/2017 10:53 AM - sorah (Sorah Fukumori)

Status:	Closed
Priority:	Normal
Assignee:	sorah (Sorah Fukumori)
Target version:	

Description

By using File::TMPFILE (O_TMPFILE) allows us to create a file without directory entries.

While open(2) with O_TMPFILE don't create a file without directory entries, it still requires a directory name to determine a file system to create a file.

Current Ruby implementation holds such directory names in fptr->pathv and retrievable via File#path.
But such paths are useless and may raise errors. For example, some code 1 checks File#path availability then when available, it attempts to use the path to open a file in different fd, finally raises Errno::EISDIR.

This patch changes File#path (fptr->pathv) not to return String if a fd is opened with O_TMPFILE.

Related issues:

Related to Ruby trunk - Feature #13577: Digest.file accidentally receives Fil...	**Assigned**

Associated revisions

Revision efd36678 - 05/15/2017 12:18 PM - sorah (Sorah Fukumori)

[DOC] File#path result can be inaccurate

- file.c(rb_file_path): [DOC] Note that the pathname returned by this method can be inaccurate, for instance file gets moved, renamed, deleted or is created with File::TMPFILE option.

Relates to [Feature #13568]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@58734 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 58734 - 05/15/2017 12:18 PM - sorah (Sorah Fukumori)

[DOC] File#path result can be inaccurate

- file.c(rb_file_path): [DOC] Note that the pathname returned by this method can be inaccurate, for instance file gets moved, renamed, deleted or is created with File::TMPFILE option.

Relates to [Feature #13568]

Revision 58734 - 05/15/2017 12:18 PM - sorah (Sorah Fukumori)

[DOC] File#path result can be inaccurate

- file.c(rb_file_path): [DOC] Note that the pathname returned by this method can be inaccurate, for instance file gets moved, renamed, deleted or is created with File::TMPFILE option.

Relates to [Feature #13568]

Revision 58734 - 05/15/2017 12:18 PM - sorah (Sorah Fukumori)

[DOC] File#path result can be inaccurate

- file.c(rb_file_path): [DOC] Note that the pathname returned by this method can be inaccurate, for instance file gets moved, renamed, deleted or is created with File::TMPFILE option.

Relates to [Feature #13568]

Revision 75cda5e2 - 08/31/2017 11:14 AM - sorah (Sorah Fukumori)

File#path: Raise IOError when a file is O_TMPFILE

File#path for a file opened with O_TMPFILE has no meaning.

A filepath returned by this method isn't guarranteed about its accuracy, but files opened with O_TMPFILE are known its recorded path has no meaning. So let them not to return any pathname.

After a discussion in ruby-core, just returning Qnil makes guessing the root cause difficult. Instead, this patch makes the method to raise an error.

Other consideration is calling fnctl(2) on rb_file_path, but it adds a overhead, and it's difficult to determine O_TMPFILE status after fd has been closed.

[Feature #13568]

- io.c(rb_file_open_generic): Set Qnil to fptr->pathv when opening a file using O_TMPFILE

- file.c(rb_file_path): Raise IOError when fptr->pathv is Qnil

- file.c(rb_file_path): [DOC] Update for the new behavior

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@59704 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 59704 - 08/31/2017 11:14 AM - sorah (Sorah Fukumori)

File#path: Raise IOError when a file is O_TMPFILE

File#path for a file opened with O_TMPFILE has no meaning.

A filepath returned by this method isn't guarranteed about its accuracy, but files opened with O_TMPFILE are known its recorded path has no meaning. So let them not to return any pathname.

After a discussion in ruby-core, just returning Qnil makes guessing the root cause difficult. Instead, this patch makes the method to raise an error.

Other consideration is calling fnctl(2) on rb_file_path, but it adds a overhead, and it's difficult to determine O_TMPFILE status after fd has been closed.

[Feature #13568]

- io.c(rb_file_open_generic): Set Qnil to fptr->pathv when opening a file using O_TMPFILE

- file.c(rb_file_path): Raise IOError when fptr->pathv is Qnil

- file.c(rb_file_path): [DOC] Update for the new behavior

Revision 59704 - 08/31/2017 11:14 AM - sorah (Sorah Fukumori)

File#path: Raise IOError when a file is O_TMPFILE

File#path for a file opened with O_TMPFILE has no meaning.

A filepath returned by this method isn't guarranteed about its accuracy, but files opened with O_TMPFILE are known its recorded path has no meaning. So let them not to return any pathname.

After a discussion in ruby-core, just returning Qnil makes guessing the root cause difficult. Instead, this patch makes the method to raise an error.

Other consideration is calling fnctl(2) on rb_file_path, but it adds a overhead, and it's difficult to determine O_TMPFILE status after fd has been closed.

[Feature #13568]

- io.c(rb_file_open_generic): Set Qnil to fptr->pathv when opening a file using O_TMPFILE

- file.c(rb_file_path): Raise IOError when fptr->pathv is Qnil

- file.c(rb_file_path): [DOC] Update for the new behavior

Revision 59704 - 08/31/2017 11:14 AM - sorah (Sorah Fukumori)

File#path: Raise IOError when a file is O_TMPFILE

File#path for a file opened with O_TMPFILE has no meaning.

A filepath returned by this method isn't guarranteed about its accuracy,
but files opened with O_TMPFILE are known its recorded path has no
meaning. So let them not to return any pathname.

After a discussion in ruby-core, just returning Qnil makes guessing the
root cause difficult. Instead, this patch makes the method to raise an
error.

Other consideration is calling fnctl(2) on rb_file_path, but it adds a
overhead, and it's difficult to determine O_TMPFILE status after fd has
been closed.

[Feature #13568]

- io.c(rb_file_open_generic): Set Qnil to fptr->pathv when opening a
  file using O_TMPFILE

- file.c(rb_file_path): Raise IOError when fptr->pathv is Qnil

- file.c(rb_file_path): [DOC] Update for the new behavior

Revision de6b788f - 08/31/2017 11:22 AM - sorah (Sorah Fukumori)

add NEWS entry for [Feature #13568] r59704

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@59705 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 59705 - 08/31/2017 11:22 AM - sorah (Sorah Fukumori)

add NEWS entry for [Feature #13568] r59704

Revision 59705 - 08/31/2017 11:22 AM - sorah (Sorah Fukumori)

add NEWS entry for [Feature #13568] r59704

Revision 59705 - 08/31/2017 11:22 AM - sorah (Sorah Fukumori)

add NEWS entry for [Feature #13568] r59704

History

#1 - 05/15/2017 11:44 AM - Hanmac (Hans Mackowiak)

not pro or against it but what do you think of that part?

```
#ifdef O_TMPFILE
    if (!(oflags & O_TMPFILE))
#endif
    {
        fptr->pathv = pathv;
    }
```

you might leaveout the {} but i think its more readable that way

#2 - 05/15/2017 11:45 AM - sorah (Sorah Fukumori)

Let me explain about the background bit detail.

The example I put in the previous post https://github.com/aws/aws-sdk-ruby/blob/v2.9.17/aws-sdk-core/lib/aws-sdk-core/checksums.rb#L15 is from aws-sdk-ruby gem. The core part of this example is like:

```
require 'digest/sha2'
```

```
def upload(body)
  checksum = if body.kind_of?(IO) && body.path
    Digest::SHA256.file(body.path)
  else
    # ...
  end
  # ...
end
```

As a user of this library, I can head the Errno::EISDIR like the following code using O_TMPFILE:

```
File.open('/tmp', File::RDWR | File::TMPFILE) do |io|
  upload io #=> Errno::EISDIR
end
```

AFAIK There's no way to determine a File object is made with O_TMPFILE.

Similar situations (path given by File#path doesn't exist or is not valid) also can be made by deleting file before closing fd. But path in File objects made by O_TMPFILE is completely sure that it doesn't point a file of the object itself, so I think returning String is meaningless from such objects.

#3 - 05/15/2017 11:47 AM - sorah (Sorah Fukumori)

Hanmac (Hans Mackowiak) wrote:

> not pro or against it but what do you think of that part?

I don't have strong opinion on the style for lines you pointed out...

#4 - 05/15/2017 12:01 PM - Hanmac (Hans Mackowiak)

i see your change and did read the man page of O_TMPFILE
to check if there is any useful path this could return. (there isn't)

now i am pro this change.
does ruby has a way to use linkat like to seen there?
http://man7.org/linux/man-pages/man2/open.2.html

hm about Tempfile, does it already use O_TMPFILE internal, or can it maybe changed to that?
(might probably for another ticket if not already done)

as for my code change, i thought it would be better if you don't have a duplicated line in there

#5 - 05/15/2017 12:19 PM - sorah (Sorah Fukumori)

- Status changed from Open to Closed

Applied in changeset trunk|r58734.

[DOC] File#path result can be inaccurate

- file.c(rb_file_path): [DOC] Note that the pathname returned by this method can be inaccurate, for instance file gets moved, renamed, deleted or is created with File::TMPFILE option.

Relates to [Feature #13568]

#6 - 05/15/2017 12:19 PM - sorah (Sorah Fukumori)

- Status changed from Closed to Open

Reopening due to closed via r58734, a document change.

#7 - 05/15/2017 12:22 PM - sorah (Sorah Fukumori)

Hanmac (Hans Mackowiak) wrote:

> now i am pro this change.
> does ruby has a way to use linkat like to seen there?
> http://man7.org/linux/man-pages/man2/open.2.html

I think we don't have one yet. But I'm not positive to give side effect on File#path.

> hm about Tempfile, does it already use O_TMPFILE internal, or can it maybe changed to that?
> (might probably for another ticket if not already done)

It's separate issue.

#8 - 05/15/2017 12:25 PM - sorah (Sorah Fukumori)

Few additional things pointed out by some conversations:

- We miss a way to retrieve a directory path used to create tmpfile
  - Although file system can be identified by File#stat
- Digest::*.file don't expect a File object, but it's actually working due to File.open calls to_path internally

#9 - 05/15/2017 01:25 PM - Hanmac (Hans Mackowiak)

yeah your points should be done in extra issues i think.

for Digest i think it should first check if its an (direct) instant of IO, or has a to_io method.
if yes then it should read the file directly instead of trying to open with a new File.open

#10 - 05/15/2017 02:18 PM - sorah (Sorah Fukumori)

fix proposed for aws-sdk-ruby gem https://github.com/aws/aws-sdk-ruby/pull/1516 but I still think returning nil from File#path when path is surely known of its unavailability is happier than current behavior.

#11 - 05/15/2017 02:43 PM - Hanmac (Hans Mackowiak)

yeah your fix for aws does looks good, but i think that should be done in Digest directly if able. (for another ticket)

#12 - 05/15/2017 04:32 PM - sorah (Sorah Fukumori)

Agreed, but my PR sent to aws-sdk-ruby includes different fix to remove incorrect usage of File#path, so it's necessary.

#13 - 05/19/2017 09:17 AM - sorah (Sorah Fukumori)

- Assignee set to sorah (Sorah Fukumori)

- Status changed from Open to Assigned

We discussed about this today in DevelopersMeeting20170519Japan. Conclution in this meeting is:

- Return unexist path with informational message, like in Linux procfs symbolic link (/proc/PID/fd/N → /tmp/#165106976 (deleted)).
- Other considerations:
  - Returning nil has no traceability.
  - Raising error is an alternate way, but it doesn't sound good.

Please see the published log for conversation details in the meeting.

#14 - 05/19/2017 11:15 AM - Hanmac (Hans Mackowiak)

i think returing an unexisting path whould be way problematic if it isnt checked if it exist.
other functions might depend on it.
and trying to open "/tmp/#165106976 (deleted)" in (read)/writing mode might result in unexpected results.

can it later checked if a file was open with O_TMPFILE?
if not, when with unexisting path, you can't know if the file can be reopen.
that might be related to #13576

i would prefer it to return nil.

PS: there might be ways to get the filesystem without using that File#path method, so return nil should be okay

#15 - 05/20/2017 02:49 AM - shyouhei (Shyouhei Urabe)

At the meeting we discussed the use case of File#path and we thought there are 2 kinds:

- Pass it to open()
- Pass it to printf() or variant

We considered both and concluded it is best to return a nonexistent path, not nil.

Hanmac (Hans Mackowiak) wrote:

> i think returing an unexisting path whould be way problematic if it isnt checked if it exist.

> other functions might depend on it.
> and trying to open "/tmp/#165106976 (deleted)" in (read)/writing mode might result in unexpected results.

Trying to open nil is like this:

```
irb(main):001:0> open(nil)
TypeError: no implicit conversion of nil into String
        from (irb):1:in `open'
```

OTOH trying to open nonexistent path is:

```
irb(main):001:0> open("/tmp/#165106976 (deleted)")
Errno::ENOENT: No such file or directory @ rb_sysopen - /tmp/#165106976 (deleted)
        from (irb):1:in `initialize'
        from (irb):1:in `open'
```

Do you think nil is better? I think nonexistent path is far more descriptive.

> can it later checked if a file was open with O_TMPFILE?
> if not, when with unexisting path, you can't know if the file can be reopen.
> that might be related to #13576

Yes. We found that quite generally speaking, it is a bad idea to pass a file's path into open. Because such thing is not guaranteed to exist. I proposed prohibiting File#path but that was rejected; because path is not only meant to be passed to open but also for inspection. When passing to printf or logger or something like that, something other than nil is desirable. I filed #13576 to focus on "file's path passed to open" situation.

> i would prefer it to return nil.
>
> PS: there might be ways to get the filesystem without using that File#path method, so return nil should be okay

(Out of curiousity) are there such ways? What do you have in mind?

#16 - 05/20/2017 07:20 AM - Hanmac (Hans Mackowiak)

like i said, open with write mode is the problem.
because it didn't fail with the nonexistent path

an way for it to make it work for printf or logger would be to make it's #inspect working (with not using path in that case)

about the filesystem, can #stat be used for something like that?

#17 - 05/20/2017 12:51 PM - nobu (Nobuyoshi Nakada)

shyouhei (Shyouhei Urabe) wrote:

> At the meeting we discussed the use case of File#path and we thought there are 2 kinds:
>
> - Pass it to open()
> - Pass it to printf() or variant

A correction; The method which open calls is to_path, not path.
These are different.

> Do you think nil is better? I think nonexistent path is far more descriptive.

The write mode problem is a matter.
It would silently (and wrongly) success to write.

I think that File#to_path should raise if the path dost not exist, but untouch File#path.

#18 - 05/20/2017 06:48 PM - naruse (Yui NARUSE)

- Related to Feature #13577: Digest.file accidentally receives File object but uses file path added

#19 - 05/22/2017 05:01 AM - shyouhei (Shyouhei Urabe)

nobu (Nobuyoshi Nakada) wrote:

> shyouhei (Shyouhei Urabe) wrote:

> > At the meeting we discussed the use case of File#path and we thought there are 2 kinds:
> >
> > - Pass it to open()
> > - Pass it to printf() or variant
>
> A correction; The method which open calls is to_path, not path.
> These are different.

Yes, I'm talking about File#path. Situations like ruby-core:81167 is in my mind. #to_path is a separate issue.

> > Do you think nil is better? I think nonexistent path is far more descriptive.
>
> The write mode problem is a matter.
> It would silently (and wrongly) success to write.

What about adding a slash then? like:

```
irb(main):001:0> open("/tmp/#165106976 (deleted / nonexistent)", "w")
Errno::ENOENT: No such file or directory @ rb_sysopen - /tmp/#165106976 (deleted / nonexistent)
        from (irb):1:in `initialize'
        from (irb):1:in `open'
```

> I think that File#to_path should raise if the path dost not exist, but untouch File#path.

For #to_path, go to #13576. This issue is about #path, and I think adding description is the best known solution; leave it untouched does not save anything.

#20 - 05/22/2017 07:49 AM - nobu (Nobuyoshi Nakada)

shyouhei (Shyouhei Urabe) wrote:

> nobu (Nobuyoshi Nakada) wrote:
>
> > A correction; The method which open calls is to_path, not path.
> > These are different.
>
> Yes, I'm talking about File#path. Situations like ruby-core:81167 is in my mind. #to_path is a separate issue.

File#path is not relevant to that situation at all.
Digest.file just opens the argument, and open calls to_path method on it to implicit conversion.

```
$ ruby -rdigest -e 'obj = Object.new; def obj.path;raise "path!"; end; Digest::MD5.file(obj)'
Traceback (most recent call last):
    4: from -e:1:in `<main>'
    3: from /opt/local/lib/ruby/2.5.0/digest.rb:35:in `file'
    2: from /opt/local/lib/ruby/2.5.0/digest.rb:50:in `file'
    1: from /opt/local/lib/ruby/2.5.0/digest.rb:50:in `open'
/opt/local/lib/ruby/2.5.0/digest.rb:50:in `initialize
': no implicit conversion of Object into String (TypeError)

$ ruby -rdigest -e 'obj = Object.new; def obj.to_path;raise "to_path!"; end; Digest::MD5.file(obj)'
Traceback (most recent call last):
    5: from -e:1:in `<main>'
    4: from /opt/local/lib/ruby/2.5.0/digest.rb:35:in `file'
    3: from /opt/local/lib/ruby/2.5.0/digest.rb:50:in `file'
    2: from /opt/local/lib/ruby/2.5.0/digest.rb:50:in `open'
    1: from /opt/local/lib/ruby/2.5.0/digest.rb:50:in `initialize'
-e:1:in `to_path': to_path! (RuntimeError)
```

> > > Do you think nil is better? I think nonexistent path is far more descriptive.
> >
> > The write mode problem is a matter.
> > It would silently (and wrongly) success to write.
>
> What about adding a slash then? like:

```
irb(main):001:0> open("/tmp/#165106976 (deleted / nonexistent)", "w")
Errno::ENOENT: No such file or directory @ rb_sysopen - /tmp/#165106976 (deleted / nonexistent)
        from (irb):1:in `initialize'
        from (irb):1:in `open'
```

A dedicated exception such as "Unnamed temporary file in some path" from to_path would be more descriptive and better, I think.

> > I think that File#to_path should raise if the path dost not exist, but untouch File#path.
>
> For #to_path, go to #13576. This issue is about #path, and I think adding description is the best known solution; leave it untouched does not save anything.

I think this issue is false assertion at the first.

#21 - 05/22/2017 08:17 AM - nobu (Nobuyoshi Nakada)

```
$ ./x86_64-linux/bin/ruby -e 'f = open(File.expand_path("/tmp"), File::RDWR|File::TMPFILE); p open(f).read'
Traceback (most recent call last):
    2: from -e:1:in `<main>'
    1: from -e:1:in `open'
-e:1:in `to_path': unnamed temporary file at /tmp (IOError)
```

```
diff --git a/file.c b/file.c
index b234910d5d..261997ecb2 100644
--- a/file.c
+++ b/file.c
@@ -367,7 +367,6 @@ apply2files(void (*func)(const char *, VALUE, void *), int argc, VALUE *argv, vo
 /*
  *  call-seq:
  *     file.path  ->  filename
- *     file.to_path  ->  filename
  *
  *  Returns the pathname used to create <i>file</i> as a string. Does
  *  not normalize the name.
@@ -391,6 +390,38 @@ rb_file_path(VALUE obj)
     return rb_obj_taint(rb_str_dup(fptr->pathv));
 }

+#ifdef O_TMPFILE
+/*
+ *  call-seq:
+ *     file.to_path  ->  filename
+ *
+ *  Returns the pathname to open the <i>file</i> as a string.
+ *
+ *  The pathname may not point the file corresponding to <i>file</i>.
+ *  e.g. file has been moved, deleted, or created with <code>File::TMPFILE</code> option.
+ */
+
+static VALUE
+rb_file_to_path(VALUE obj)
+{
+    rb_io_t *fptr;
+    int flags;
+
+    fptr = RFILE(rb_io_taint_check(obj))->fptr;
+    rb_io_check_initialized(fptr);
+    flags = fcntl(fptr->fd, F_GETFL);
+    if (flags == -1) rb_sys_fail_path(fptr->pathv);
+    if ((flags & O_TMPFILE) == O_TMPFILE) {
+   rb_raise(rb_eIOError, "unnamed temporary file at %"PRIsVALUE,
+        fptr->pathv);
+    }
+    if (NIL_P(fptr->pathv)) return Qnil;
+    return rb_obj_taint(rb_str_dup(fptr->pathv));
+}
+#else
+# define rb_file_to_path rb_file_path
+#endif
+
 static size_t
 stat_memsize(const void *p)
 {
@@ -6140,7 +6171,7 @@ Init_File(void)
     rb_define_const(rb_mFConst, "NULL", rb_fstring_cstr(null_device));

     rb_define_method(rb_cFile, "path",  rb_file_path, 0);
-    rb_define_method(rb_cFile, "to_path",  rb_file_path, 0);
+    rb_define_method(rb_cFile, "to_path",  rb_file_to_path, 0);
     rb_define_global_function("test", rb_f_test, -1);

     rb_cStat = rb_define_class_under(rb_cFile, "Stat", rb_cObject);
```

#22 - 05/22/2017 08:26 AM - shyouhei (Shyouhei Urabe)

nobu (Nobuyoshi Nakada) wrote:

> ```
> $ ./x86_64-linux/bin/ruby -e
> 'f = open(File.expand_path("/tmp"), File::RDWR|File::TMPFILE); p open(f).read'
> Traceback (most recent call last):
> 	2: from -e:1:in `<main>'
> 	1: from -e:1:in `open'
> -e:1:in `to_path': unnamed temporary file at /tmp (IOError)
> ```

Why are you ignoring #13576, after I repeatedly mentioned in this thread?

#23 - 05/22/2017 09:26 AM - duerst (Martin Dürst)

- Subject changed from File#path for O_TMPFILE fds are unmeaning to File#path for O_TMPFILE fds has no meaning

#24 - 05/22/2017 11:25 AM - nobu (Nobuyoshi Nakada)

Once #13576 is introduced, this issue becomes stale.
It doesn't make sense to change File#path.

#25 - 05/22/2017 11:36 AM - shyouhei (Shyouhei Urabe)

nobu (Nobuyoshi Nakada) wrote:

> Once #13576 is introduced, this issue becomes stale.
> It doesn't make sense to change File#path.

OK, I see your opinion over File#path. I have different opinion though. I agree with sorah here; a file created using O_TMPFILE can never have a meaningful path.

#26 - 05/23/2017 05:47 PM - Eregon (Benoit Daloze)

I think an exception is the best way to be clear about the problem here, nil or a fake path are most likely to raise exceptions, but later and in a harder to debug way.
Moreover, this only happens for O_TMPFILE files, so it seems reasonable to have different behavior.

#27 - 08/31/2017 06:05 AM - matz (Yukihiro Matsumoto)

I vote for raising exceptions (ENOENT?).

Matz.

#28 - 08/31/2017 07:04 AM - sorah (Sorah Fukumori)

- File 0001-File-path-Raise-IOError-when-a-file-is-O_TMPFILE.patch added

Nobu's patch at [ruby-core:81329] fails when fd is closed... Updated the patch.

- Raise IOError when fptr->pathv is Qnil on File#path
- Set Qnil to fptr->pathv when opening file with O_TMPFILE
- File#to_path and FIle#path behave same

#29 - 08/31/2017 11:14 AM - sorah (Sorah Fukumori)

- Status changed from Assigned to Closed

Applied in changeset trunk|r59704.

File#path: Raise IOError when a file is O_TMPFILE

File#path for a file opened with O_TMPFILE has no meaning.

A filepath returned by this method isn't guarranteed about its accuracy, but files opened with O_TMPFILE are known its recorded path has no meaning. So let them not to return any pathname.

After a discussion in ruby-core, just returning Qnil makes guessing the root cause difficult. Instead, this patch makes the method to raise an error.

Other consideration is calling fnctl(2) on rb_file_path, but it adds a overhead, and it's difficult to determine O_TMPFILE status after fd has been closed.

[Feature #13568]

- io.c(rb_file_open_generic): Set Qnil to fptr->pathv when opening a file using O_TMPFILE

- file.c(rb_file_path): Raise IOError when fptr->pathv is Qnil

- file.c(rb_file_path): [DOC] Update for the new behavior

Files

tmpfile-path.patch	1.96 KB	05/15/2017	sorah (Sorah Fukumori)
0001-File-path-Raise-IOError-when-a-file-is-O_TMPFILE.patch	3.7 KB	08/31/2017	sorah (Sorah Fukumori)